Differences in Floating-Point Arithmetic Between Intel® Xeon® Processors and the Intel® Xeon Phi™ Coprocessor

The floating-point model used by the Intel® Compiler and its application to Intel® Xeon® processors is described in the paper “Consistency of Floating-Point Results using the Intel® Compiler.” For a suitable choice of settings, the compiler generates code that is fully compliant with the ANSI language standards and the IEEE-754 standard for binary floating-point arithmetic. Compiler options give the user control over the tradeoffs between optimizations for performance, accuracy, reproducibility of results and strict conformance with these standards.

The same floating-point model applies to the Intel® Xeon Phi™ coprocessor, but the architectural differences compared to Intel Xeon processors lead to a few small differences in implementation. Those differences are the subject of this paper.
**Basics**

The Intel Xeon Phi coprocessor supports the same floating-point data types as the Intel Xeon processor. Single (32-bit) and double (64-bit) precision are supported in hardware; quadruple (128-bit) precision is supported through software. Extended (80-bit) precision is supported through the x87 instruction set. Denormalized numbers and gradual underflow are supported, but abrupt underflow is the default at all optimization levels except -O0. The same set of rounding modes is supported as for Intel Xeon processors.

**Floating-Point Exceptions**

The biggest differences arise in the treatment of floating-point exceptions. The vector floating-point unit on the Intel Xeon Phi coprocessor flags but does not support trapping of floating-point exceptions. The corresponding bit in the VXCSR register is protected; attempts to modify it result in a segmentation fault. Some compiler options such as -fp-trap (C/C++) or -fpe0 (Fortran) that would unmask floating-point exceptions on Intel Xeon processors are unsupported on Intel Many Integrated Core Architecture (Intel® MIC architecture).

The options -fp-model except or -fp-model strict still request strict, standard-conforming semantics for floating-point exceptions. This is achieved by generating x87 code for floating-point operations instead of code that makes use of Intel® Initial Many Core Instructions (Intel® IMCI). Because such code cannot be vectorized, this may have a substantial impact on performance. Nevertheless, these options may be useful for application debugging. For similar reasons, the options -ansi and -fmath-errno may result in calls to math functions that are implemented using x87 rather than Intel® IMCI.

In the Fortran compiler 13.0, the IEEE_FEATURES, IEEE_ARITHMETIC and IEEE_EXCEPTIONS modules are not yet updated for the properties of the Intel Xeon Phi coprocessor.

**The -fp-model switch**

The same settings of the -fp-model switch are supported as for Intel Xeon processors. The default setting is -fp-model fast=1 for both. The behavior of the -fp-model precise option is the same, though the impact on performance may be somewhat greater for Intel Xeon Phi coprocessors, because of the larger vector width on Intel MIC architecture and larger potential performance gain from the vectorization of reduction loops and loops containing transcendental math functions, square roots or division. The impact of -fp-model except and -fp-model strict on performance may be greater, for the reasons noted above.
The setting –fp-model fast=2 sets the –fimf-domain-exclusion switch and enables faster, inlined versions of some math functions for the Intel Xeon Phi Coprocessor, see the section “Precision of Math Functions” below.

The –fp-model precise switch enables arithmetic using denormalized numbers and disables abrupt underflow; abrupt [gradual] underflow can be enabled (but not required) explicitly using -[no-]ftz. This behavior is the same as on Intel Xeon processors.

**Fused Multiply-Add**

Intel Xeon processors up to and including Intel® 3rd Generation Core™ processors do not have a fused multiply-add (FMA) instruction. The FMA instruction on Intel Xeon Phi coprocessors only performs a single rounding on the final result, so can yield results that differ very slightly from separate addition and multiplication instructions.

In principle, the –fp-model strict switch would disable fused multiply-add (FMA) instructions. But since, as noted above, -fp-model strict suppresses Intel® IMCI in favor of legacy x87 floating-point instructions, this additional behavior is moot. FMA operations are enabled by default, but may be disabled directly by the switch –no-fma. FMA operations are not disabled by –fp-model precise.

**Precision of Math Functions**

In the Intel® Composer XE 2013 product, the compiler for Intel Xeon Phi coprocessors invokes medium accuracy (< 4 ulp) transcendental functions for both scalar and vector code by default, mostly as calls to libsvml. For Intel Xeon processors, the default is libm (< 0.55 ulp) for scalar code and medium accuracy libsvml (< 4 ulp) for vector code. On Intel Xeon Phi coprocessors, division defaults to medium accuracy inlined code that uses a reciprocal instruction, whereas in the initial 13.0 compiler release, square roots call a medium accuracy SVML function by default (will change to inlined code in a forthcoming update). An inlined version of square root may be obtained with -fimf-domain-exclusion=15:sqrt (double precision) or -fimf-domain-exclusion=15:sqrt (single precision). See compiler documentation for details of the excluded domains. The –fimf-domain-exclusion switch may also be used to obtain inlined versions of certain other math functions. High accuracy (typically 0.6
ulp) vectorizable SVML versions of divide, square root and transcendental functions may be obtained with –fimf-precision=high.

-fp-model precise results in high accuracy (<0.55 ulp), scalar calls to libm for transcendents and to libsvml for square roots. For division, it results in an x87 division instruction. The switches –no-fast- transcendents, –prec-sqrt and –prec-div respectively have the same effect. In forthcoming updates, vectorizable inlined code sequences using Intel® IMCI may be used for division and square roots in preference to x87 instructions for these switches and for –fp-model precise. x87 instructions will continue to be generated for –fp-model strict.


Comparing floating-point results between Intel Xeon Phi coprocessors and Intel Xeon Processors

In general, floating-point computations on an Intel Xeon Phi coprocessor may not give bit-for-bit identical results to the equivalent computations on an Intel Xeon processor, even though underlying hardware instructions conform to the same standards. Compiler optimizations may be implemented differently, math functions may be implemented differently, etc. The fused multiply-add (FMA) instruction available on the Intel Xeon Phi coprocessor is a common source of differences. Nevertheless, the following guidelines may help to minimize differences between results obtained on different platforms, at a cost in performance:

Build your application on both platforms using –fp-model precise –fp-model source.

Build your application on the Intel Xeon Phi coprocessor with –no-fma, to disable the use of FMA instructions. [ Alternatively, for Fortran applications only, you may inhibit the use of fma instructions in individual expressions by the use of parentheses, in conjunction with the command line switch –assume protect_parens, e.g.: X = (A + (B*C)) ]. For C or C++ applications, you may disable FMA generation for individual functions by using #pragma fp_contract (off | on).

Select high accuracy math functions on both platforms, e.g. using –fimf-precision=high.

For OpenMP applications that perform reductions in parallel, set the environment variable KMP_DETERMINISTIC_REDUCTIONS=yes, use static scheduling, and use
OMP_NUM_THREADS to set the same number of threads on each platform.

For C++ applications making use of Intel® Threading Building Blocks, (Intel® TBB), the `parallel_deterministic_reduction()` function may help to obtain more consistent results on both platforms, even for differing numbers of threads.

These guidelines are intended to help enhance reproducibility and minimize variations in floating-point results between different platforms. The observed variations do not represent the true numerical uncertainty in your result, which may be much greater.
INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL’S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL’S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS’ FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or “undefined”. Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

Intel, the Intel logo, VTune, Phi and Xeon are trademarks of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others

Copyright© 2013 Intel Corporation. All rights reserved.
Performance Notice
For more complete information about performance and benchmark results, visit www.intel.com/benchmarks

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804